Use of AreaDetector and EtherCat at Diamond beamlines

Ulrik Pedersen, Jon Thompson, Tom Cobb, James Rowland, Ronaldo Mercado and Nick Rees

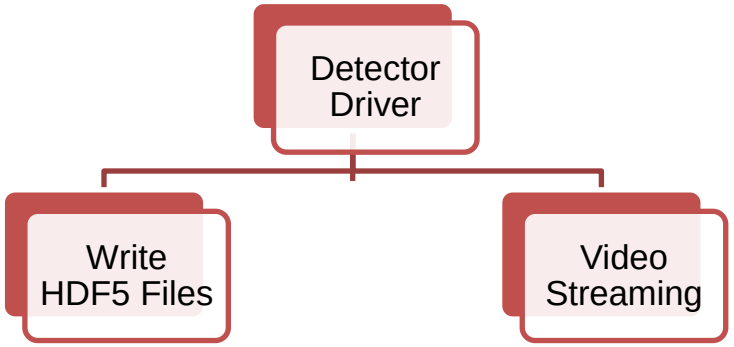
diamond

Introduction

- areaDetector
  - Background and history
  - GigE Vision detectors
  - PCO detectors
  - Spectroscopy detectors
  - Networked areaDetector and the relationship to EPICS v4
- Ethercat
  - Background and history
  - Radiation tests
  - Ethercat box I/O

Detectors and areaDetector

- areaDetector is the standard interface for detectors at Diamond.
  - It is the standard EPICS detector framework.
  - It supports most Diamond detectors.
  - It contains “plugins” - code modules that can be wired together to define a processing workflow.
- It is highly efficient:
  - Data is kept in shared memory and is shared between plugins.
  - Plugins can run in their own threads.
  - Bandwidth throttling for balanced CPU usage



Supported detectors

- From the distribution:
  - ADSC
  - Andor
  - Bruker
  - Firewire Linux/Windows
  - mar345/MarCCD
  - Perkin-Elmer flat panel
  - Pilatus
  - Prosilica
  - PVCAM
  - Roper
  - Simulation detector
  - URL
- From Diamond
  - VG Scienta
  - GigE cameras (aravis)
  - PCO Cameras
  - Pixium flat panel
  - ... plus DLS internal detectors
    - Excalibur (medipix)
    - Merlin

PCO Camera Support

- Diamond has developed a driver that supports a number of cameras from the PCO range
  - PCO 1600, 2000, 4000
  - PCO.dimax
  - PCO.edge
- The driver relies on the PCO API, so only Windows is supported.
- Latest system supported is PCO.edge
  - 2048x2048 SCMOS detector
  - Maximum frame rate is 90 Hz
  - Uses ~80% of one hyper-threaded core (2.93GHz X5670 Processor)

Spectroscopy detectors

- Mark Rivers has developed drivers for a number of digital signal processing spectroscopy systems – notably from XIA
- Not strictly areaDetector, since they don't derive from ADBase, but they use the rest of the framework.
- We are currently developing support for Quantum Detectors Xspress3
  - http://www.quantumdetectors.com/products/xspress-3
  - approximately 10 times faster than XIA systems.
- We interface to a C API running on a Linux system.
  - C API communicates via dedicated 10 GbE with Xspress3 system
- Should be available in the new year.
  - May require some special spectroscopy plugins.

Networked areaDetector

- areaDetector has
  - Named objects with metadata (asyn Ports and asyn Parameter system).
  - Ability to pass structured data between objects efficiently
    - NDArrays which are not locked, but adhere to write once, read many times semantics and have reference counting, automatic recollection and management of free lists
    - Restricted to one machine only
- Would sometimes like to pass data between machines to, for example:
  - Move data from Windows detector system to Linux
    - direct access to Linux file systems
  - Distribute data for processing
- Does this sound like EPICSv4?

Networked areaDetector

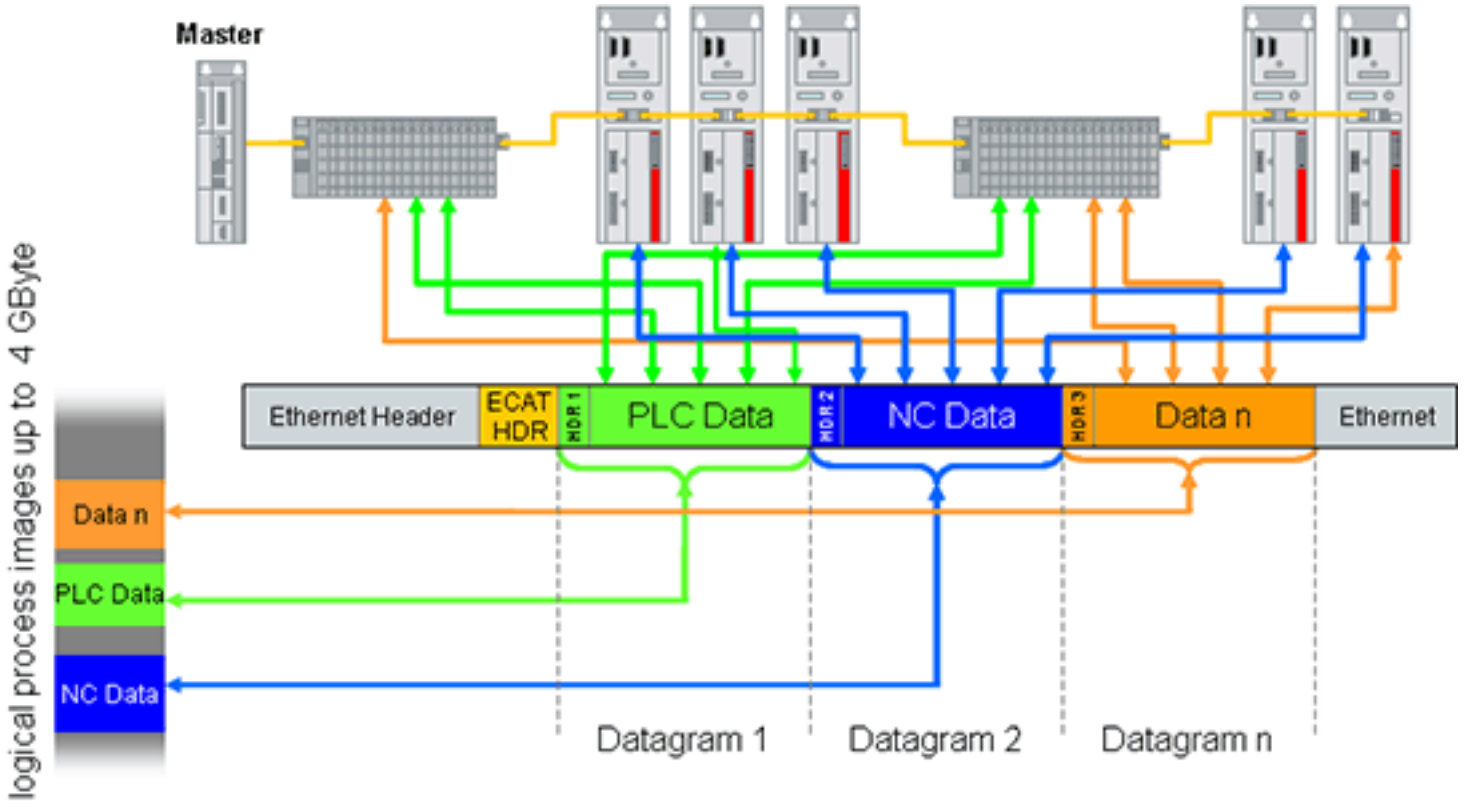
- **We are developing a producer/consumer pair of plugins to enable an area detector asyn port on one machine to be connecter to a driver port on another machine.**
- **Uses EPICSv4 pvData to serialise data for transmission and plan on using pvAccess for transmission.**
  - **Currently using pvData format but with our own simple libraries for transmission**
  - **Waiting for CAv4 to stabilise.**
- **Performance is good**
  - **Can fully saturate a 10 GbE link (~1150 MB/sec).**
  - **Can be affected by data malloc sizes on some architectures.**
  - **(We think MMU has problems managing blocks > 128k)**


EtherCAT®

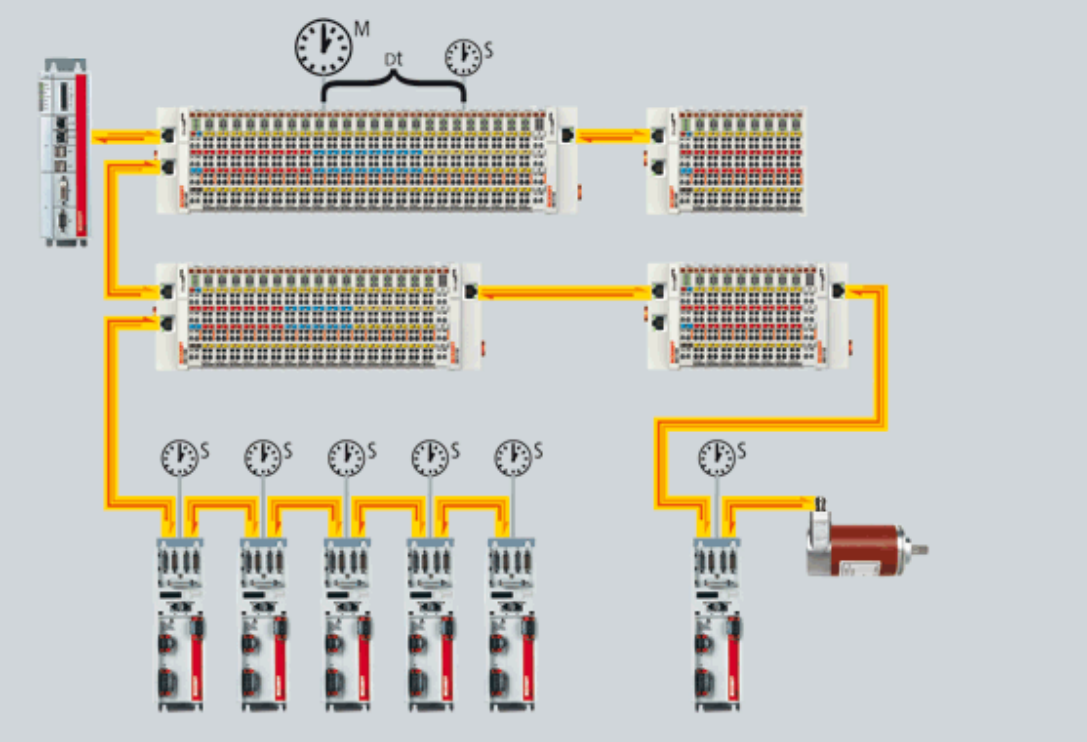
EtherCAT

- Relatively open PLC fieldbus from Beckhoff
- Open source Linux drivers exist
- No special hardware required for Fieldbus master
- High performance (if controller O/S supports it)
  - 256 digital I/O in 11 µs
  - 1000 digital I/O distributed to 100 nodes in 30 µs
  - 200 analog I/O (16 bit) in 50 µs, 20 kHz Sampling Rate
  - 100 Servo-Axis (each 8 Byte IN+OUT) in 100 µs
  - 12000 digital I/O in 350 µs
- System wide synchronisation to << 1 µs

Data multiplexed on Ethernet frame



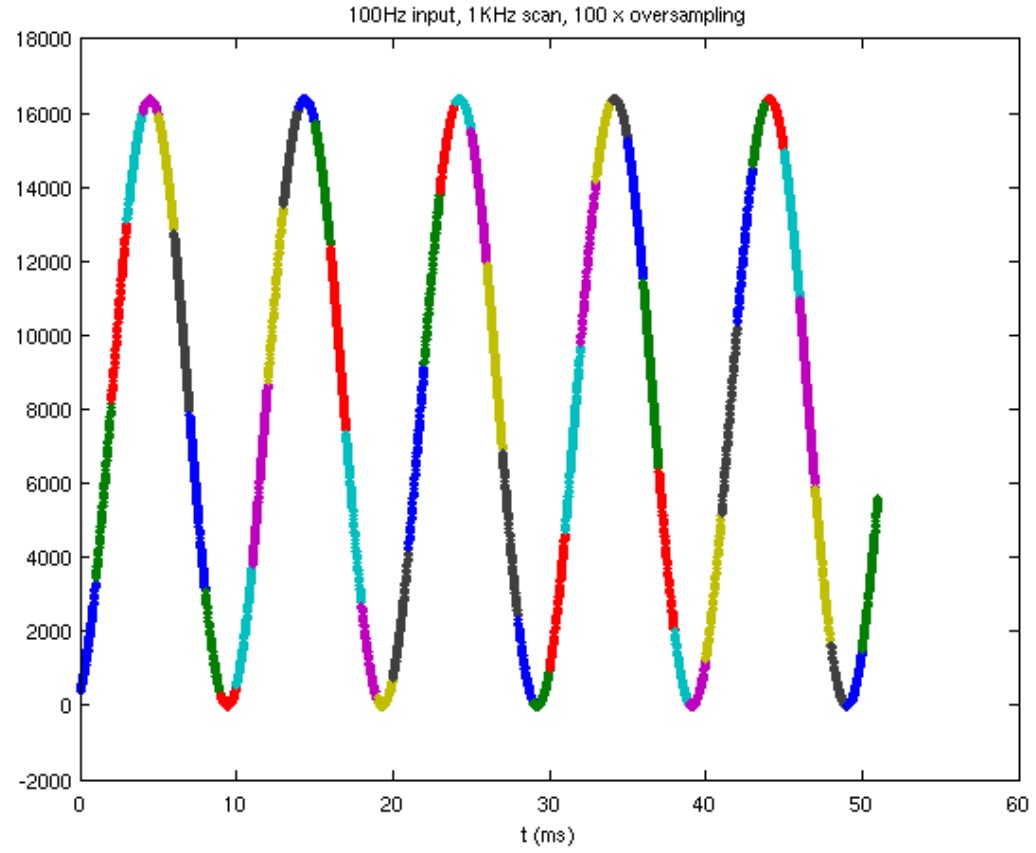
Synchronisation



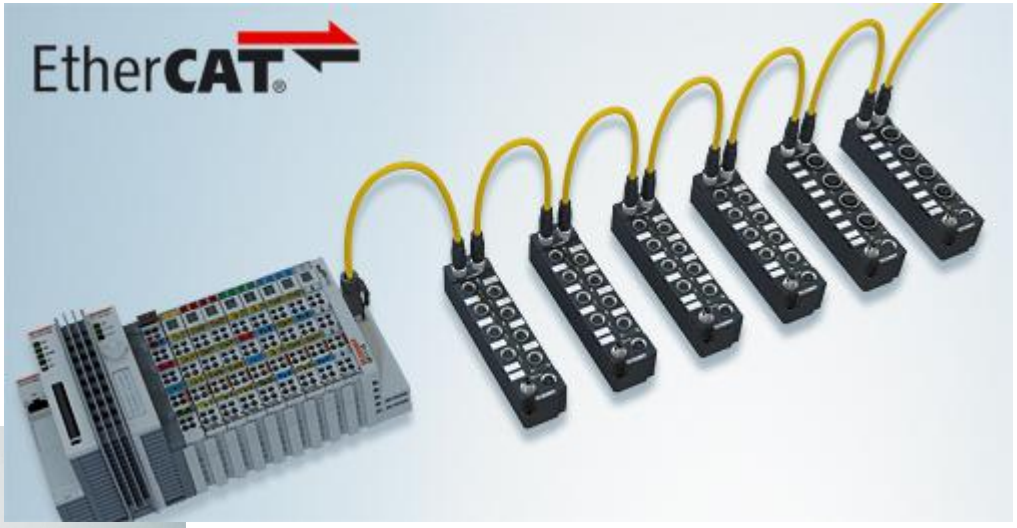
- Each EtherCAT network is actually a linear segment.
- Last device reflects the packet back to the master.
- Each device has a timer and measures the time between the outgoing and returning packet and so works out its time relative to other modules.

“Oversampling” Modules

- **Device sends N samples per bus cycle**
- **Clock PLL synchronizes acquisition with master and other devices**



Module types

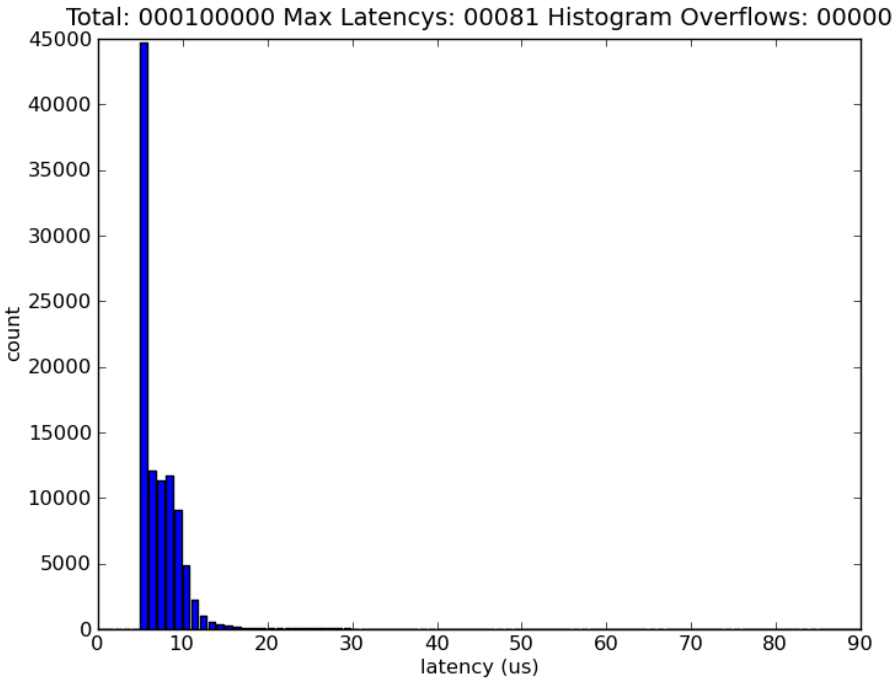


Linux host software

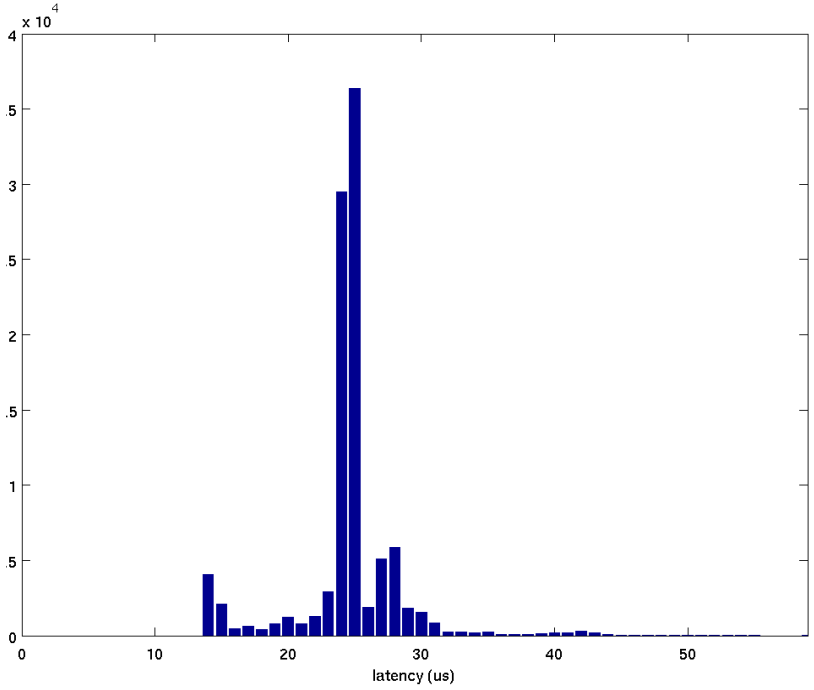
- Use Linux PREEMPT_RT kernel
  - RHEL5 MRG 2.6.24 SRPM
  - 10 μs mean, outlier of 40 μs maximum latency
  - Uses Posix calls: clock_nanosleep, mlockall, SCHED_FIFO, PRIO_INHERIT
  - Mainline Kernel absorbing patches
- We like PREEMPT_RT kernel because
  - It is closest to the main line kernel.
  - It can be used by a non-privileged user in userspace.
  - MRG realtime is RHEL6 name for package that includes RT PREEMPT
    - also includes messaging and HPC components.

PREEMPT_RT performance

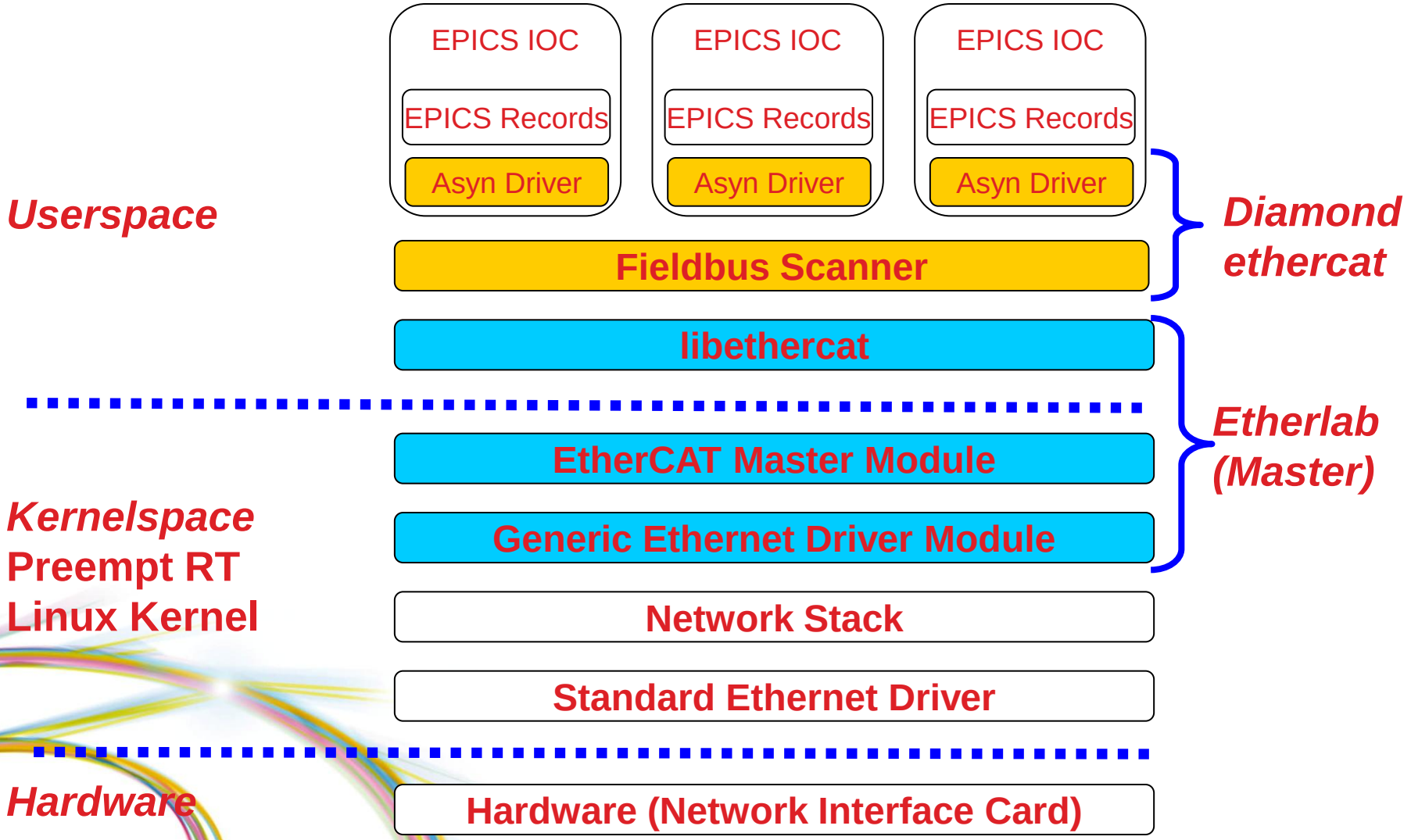
Timer latency



EtherCAT scan time



EtherCAT - Context



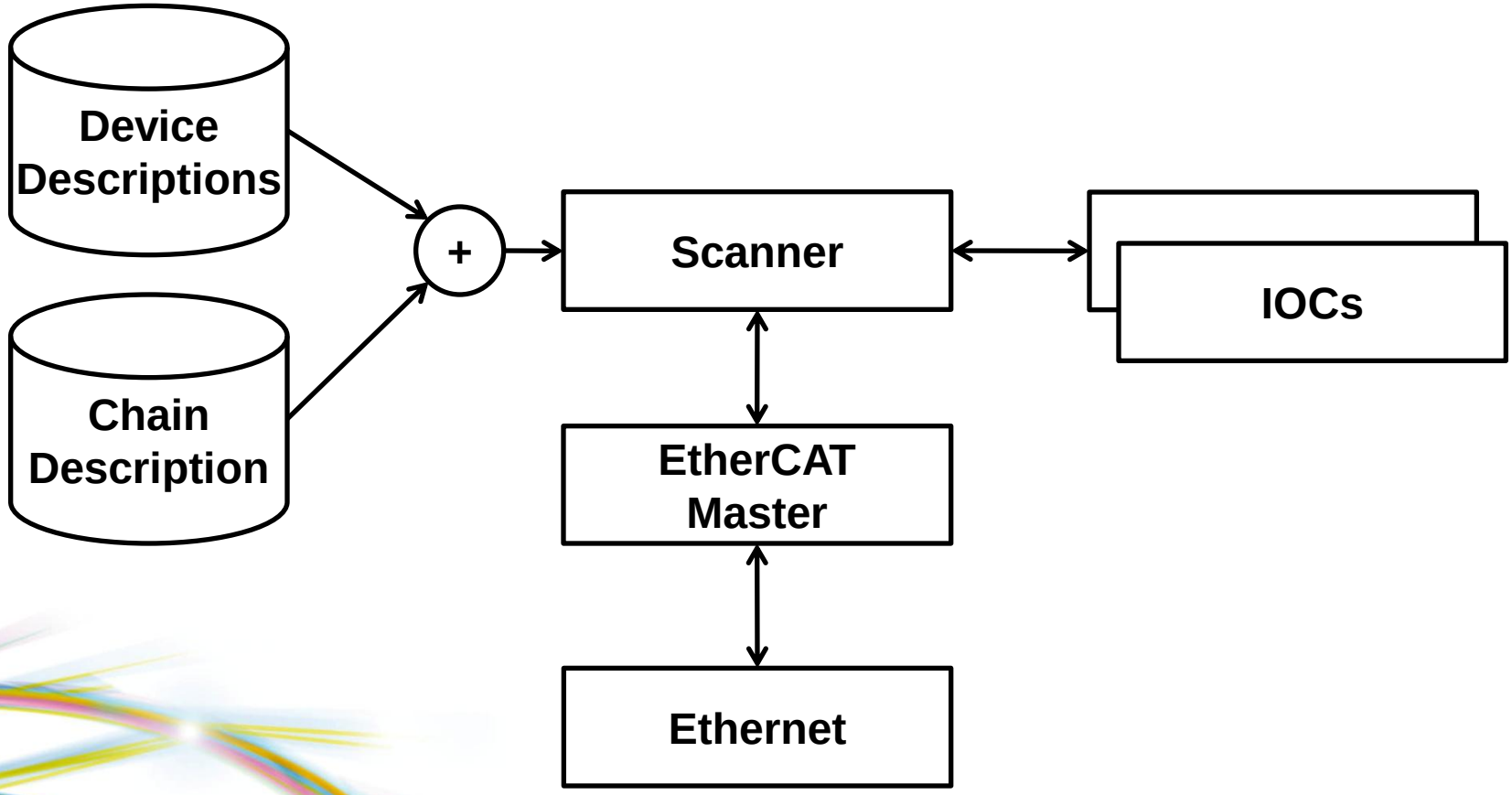
EtherCAT Master

- EtherCAT master from http://www.etherlab.org
- Kernel Driver with near identical user and kernel space APIs
- (L)GPL license.
- Has PF_PACKET generic network driver
  - some network drivers also supported explicitly.
- FMMU setup and slave state machine control
- Create your own bus scan timer and configuration files
- Uses dynamic kernel module support (DKMS) RPM
- All our development done entirely in user space with generic network driver.

Scanner

- 1KHz cyclic task
  - Cycle time can be varied.
- Pre-threaded UNIX socket server
- Reads configuration XML
- Merges writes from multiple source into a single EtherCAT transmission.
- Distributes reads from a EtherCAT response to all who are interested.

EPICS software components



EtherCat device chain XML

```
<chain>
<device type_name="EK1100" revision="0x00110000" position="0" name="PORT0" />
<device type_name="EL1014" revision="0x00100000" position="1" name="PORT1" />
<device type_name="EL1014" revision="0x00100000" DEVID="DLS123456" name="PORT2" />
<device type_name="EL1014" revision="0x00100000" position="3" name="PORT3" />
<device type_name="EL3104" revision="0x00100000" position="4" name="PORT4" />
<device type_name="EL3702" revision="0x00020000" position="5" name="PORT5" />
<device type_name="EL3702" revision="0x00020000" position="6" name="PORT6" />
<device type_name="EL3702" revision="0x00020000" position="7" name="PORT7" />
<device type_name="EL9410" revision="0x00100000" position="8" name="PORT8" />
<device type_name="EL3202-0010" revision="0x0011000a" position="9" name="PORT9" />
<device type_name="EL3202-0010" revision="0x0011000a" position="10" name="PORT10" />
</chain>
```

- Template file can be generated by scanning the installed bus with “slaveinfo” command (provide with EPICS device support).
- Device can specify device by internally stored ID or by bus position.
- EtherCAT asynDriver creates an asyn port of specified name
- asyn driver will create all the parameters specified in the vendors device type XML description.

To switch from VME I/O to EtherCAT

- **Build your EtherCAT chain.**
- **Query it using slaveinfo command and edit the resulting asyn port names to be what you want.**
  - `slaveinfo <ethernet interface>`
- **Merge this chain with the vendor device type descriptions to create full chain description file**
  - `expandChain.py <device desc dir> <chain description>`
- **Start the scanner – requires unix pipe name and the full chain description as parameters.**
  - `scanner <expanded chain file> <unix pipe name>`
- **Change database links to be:**
  - `@asyn(<PORT>) <parameter>`
- **Initialise EtherCAT driver in IOC**
  - `ecAsynInit("/tmp/scanner", 0)`

Summary

- areaDetector and the associated framework can be adapted to most detector requirements
  - speed is limited by processor, not framework
  - more and more drivers are appearing all the time
- EtherCAT is a flexible, cost effective alternative to VME
  - no vendor tie ins
  - no closed source code

Conclusions

- areaDetector
  - EPICS continues to evolve to support the needs of the experimental area.
  - areaDetector is not limited by EPICS software – it is limited by hardware.
  - areaDetector is becoming a good test bed for EPICSv4
- EtherCAT
  - Reasonably simple substitute for VME I/O for Linux
  - Good performance – sampling multiple inputs at 100 MHz is well within spec.
  - No special software or hardware needed
  - Linux real time performance is good
    - but real time EPICS on Linux is still not mainstream
  - Good performance with wide range of hardware
  - Simple to use